J. Symbolic Computation (1989) 7, 85–91

Algebraic Programming of Geometrical Calculus and Clifford Algebra

Ph. TOMBAL AND A. MOUSSIAUX

Laboratoire de Physique Mathématique, Facultés Universitaires, Rue de Bruxelles 61, 5000 *Namur, Belgium*

(*Received* 10 *November* 1987)

The basic concepts of geometrical calculus have been implemented in an algebraic programming language. In this paper, we describe how our program performs fundamental operations on multivectors.

1. Introduction

In their book Hestenes & Sobczyk (1984) are setting a new mathematical language based on Clifford calculus whose purpose is to unify various domains of algebraic and differential geometry. The main concepts are those of multivectors — which may be seen as generalisation of ordinary vectors — and of geometric product which is used to define other products such as the outer product, inner product (analog, respectively, to the exterior product and contraction for differential forms), scalar product, commutator product,...

Combinations of these products give all the well-known quantities in vector analysis, tensor calculus,... (Hsu, 1969).

In this paper, we describe one of the multiple ways to implement the concepts introduced by Hestenes & Sobczyk in a language of algebraic programmation. We chose MACSYMA because of its inherent ability to manipulate the basic elements of LISP (lists), making possible a translation of our program in any LISP-based language.

As was expected the theoretical unification has its exact counterpart in the program, since all the manipulations on multivectors reduce to a single procedure called the 'snail' function as explained in the following sections.

2. Definitions and Representation of Multivectors

A multivector can be written as the sum:

$$A = a_0 + \sum_i a^i e_i + \sum_{i_1 < i_2} a^{i_1 i_2} e_{i_1} e_{i_2} + \cdots + \sum_{i_1 < \cdots < i_r} a^{i_1 \cdots i_r} e_{i_1} \ldots e_{i_r} + \cdots$$

$$= \text{grade}\,(A, 0) + \text{grade}\,(A, 1) + \text{grade}\,(A, 2) + \cdots + \text{grade}\,(A, r) + \cdots. \quad (2.1)$$

Each term of the sum is called the grade r of A or the r-vector part of A; the first term being the scalar part (0-vector), the second term being the usual vector part (1-vector) and so on $\cdots$.

The e_i's are vectors defined such as their geometric product $e_i e_j$ follows the rules:

$$e_i e_j = -e_j e_i \quad \text{if } i \neq j, \; g_{ii} \quad \text{if } i = j, \quad \text{where } g_{ii} \text{ is a scalar.} \quad (2.2)$$

0747-7171/89/010085+7 $03.00/0

These vectors and any ordered geometric product of them form a base for the multivectors. There are as many e_i's as we want since no dimension is specified. In this way (2.1) must be seen as an infinite sum with a finite number of non-zero terms.

Since the number of terms in (2.1) is arbitrary, we decide to represent a multivector in our program by a list:

(*c*5) *aa*;

(*d*5) [@(*a*%$_s$, []), @(*a*%$_1$, [1]), @(*a*%$_2$, [2]), @(*a*%$_3$, [3]), @(*a*%$_{1,2}$, [1, 2]), @(*a*%$_{1,3}$, [1, 3]), @(*a*%$_{2,3}$, [2, 3]), @(*a*%$_{1,2,3}$, [1, 2, 3])].

Each element of the list is a function (which is called a 'snail'). This function has two arguments: the first being a coefficient which is any real algebraic expression, the second being a list of positive integers representing the geometric product of the vectors e_i's.

(An empty list stands for the scalar part). For example:

@(x, [1, 2, 3]) stands for $xe_1e_2e_3$.

The 'snail' function simplifies itself following the rules (2.2). For example:

@(x, [2, 1]) becomes @($-x$, [1, 2]) and

@(x, [1, 1, 3, 2, 2, 2]) becomes @($-xg_{11}g_{22}$, [2, 3]).

The function GRADE is used to isolate the r-vector part of the multivector. Its first argument is the name of the multivector, the second is the grade to be isolated. For example:

(*c*6) grade (*aa*, 0);

(*d*6) @(*a*%$_s$, [])

(*c*7) grade (*aa*, 1);

(*d*7) [@(*a*%$_1$, [1]), @(*a*%$_2$, [2]), @(*a*%$_3$, [3])]

(*c*8) grade (*aa*, 2);

(*d*8) [@(*a*%$_{1,2}$, [1, 2]), @(*a*%$_{1,3}$, [1, 3]), @(*a*%$_{2,3}$, [2, 3])].

3. Sums and Differences of Multivectors

The sum and difference of two multivectors are performed, respectively, by the two infix operators '++' and '--' which simply collect the coefficients of same grade and apply the corresponding operation on them. For example:

(*c*6) *bb*: multi (*b*, [1, 2]);

(*d*6) [@(*b*%$_1$, [1]), @(*b*%$_2$, [2])]

(*c*7) *bb*;

(*d*7) [@(*b*%$_1$, [1]), @(*b*%$_2$, [2])]

(*c*9) *cc*;

(*d*9) [@(*c*%$_1$, [1]), @(*c*%$_{2,3}$, [2, 3]), @(*c*%$_{1,2,3}$, [1, 2, 3])]

(*c*10) $bb++cc;$

(*d*10) $[@(c\%_1+b\%_1, [1]), @(b\%_2, [2]), @(c\%_{2,3}, [2,3]), @(c\%_{1,2,3}, [1,2,3])]$

(*c*11) $bb--cc;$

(*d*11) $[@(-c\%_1+b\%_1, [1]), @(b\%_2, [2]), @(-c\%_{2,3}, [2,3]), @(-c\%_{1,2,3}, [1,2,3])].$

4. Geometric Product

The fundamental product in geometric calculus is the geometric product from which all other kind of products are defined. By definition, the geometric product is distributive, associative and, of course, not commutative in general. Distributivity allows us to reduce the product of two multivectors to a linear combination of products of snails.

The geometric product of two snails is simply a snail whose first argument is the ordinary product of the two first arguments and whose second argument is a list made of the two last arguments appended. For example:

$$@(x, [1,2])|@(y, [1,3,5]) = @(xy, [1,2,1,3,5])$$
$$= @(-xyg_{11}, [2,3,5]).$$

The final result being automatically simplified by the snail function as explained above. The symbol $|$ is the infix operator which performs the geometric product.

A more interesting example is given by the product of two 1-vectors u and v:

(*c*14) $u;$

(*d*14) $[@(u\%_1, [1]), @(u\%_2, [2]), @(u\%_3, [3])]$

(*c*15) $v;$

(*d*15) $[@(v\%_1, [1]), @(v\%_2, [2]), @(v\%_3, [3])]$

(*c*16) $u|v;$

(*d*16) $[@(u\%_3v\%_3g_{3,3}+u\%_2v\%_2g_{2,2}+u\%_1v\%_1g_{1,1}, []), @(u\%_1v\%_2-v\%_1u\%_2, [1,2]),$
$@(u\%_1v\%_3-v\%_1u\%_3, [1,3]), @(u\%_2v\%_3-v\%_3u\%_3, [2,3])].$

We can see that the result is inhomogeneous: it involves a scalar part and a bivector part. The scalar part corresponds to the inner product $u.v$ and the bivector part corresponds to the outer product $u \wedge v$ so that uv can be written:

$$uv = u.v+u \wedge v. \tag{4.1}$$

5. Inner, Outer and Scalar Products

Formula (4.1) is a particular case of the more general formula giving the geometric product of two homogeneous multivectors A_r and B_s of grade r and grade s, respectively (Hestenes & Sobczyuk, 1984)

$$A_r|B_s = \langle A_rB_s\rangle_{|r-s|}+\langle A_rB_s\rangle_{|r-s|+2}+\cdots+\langle A_rB_s\rangle_{r+s}. \tag{5.1}$$

The inner product corresponds to the term of grade $r-s$ (grade lowering operation), the outer product is the term of grade $r+s$ (grade raising operation) so that we have the definitions (1):

$$A_r \,.\, B_s = \langle A_r B_s \rangle_{|r-s|}, \tag{5.2}$$

$$A_r \wedge B_s = \langle A_r B_s \rangle_{r+s}. \tag{5.3}$$

When $r = s = 1$ we see that (5.1) reduces to (4.1).

The usual scalar product is defined as the grade 0 of the geometric product i.e.:

$$A_r * B_s = \langle A_r B_s \rangle_0. \tag{5.4}$$

So the inner product and scalar product coincide when $r = s$.

In our program, the geometrical, inner, outer and scalar products are performed respectively by the operators $|$, ., $\sim$ and &.

Examples:

(a) The first term of (4.1) could have been computed:

(*c*17) $u.v;$

(*d*17) $@(u\%_3 v\%_3 g_{3,3} + u\%_2 v\%_2 g_{2,2} + u\%_1 v\%_1 g_{1,1}, [])$

(*c*20) $u \& v;$

(*d*20) $@(u\%_3 v\%_3 g_{3,3} + u\%_2 v\%_2 g_{2,2} + u\%_1 v\%_1 g_{1,1}, [])$

(*c*19) grade $(u|v, 0);$

(*d*19) $@(u\%_3 v\%_3 g_{3,3} + u\%_2 v\%_2 g_{g2,2} + u\%_1 v\%_1 g_{1,1}, [])$

and the second term:

(*c*18) $u \sim v;$

(*d*18) $[@(u\%_1 v\%_2 - v\%_1 u\%_2, [1,2]), @(u\%_1 v\%_3 - v\%_1 u\%_3, [1,3]),$
$@(u\%_2 v\%_3 - v\%_2 u\%_3, [2,3])]$

(*c*21) grade $(u|v, 2);$

(*d*21) $[@(u\%_1 v\%_2 - v\%_1 u\%_2, [1,2]), @(u\%_1 v\%_3 - v\%_1 u\%_3, [1,3]),$
$@(u\%_2 v\%_3 - v\%_2 u\%_3. [2,3])].$

(b) Now let's see the result of the different products of the vector u with a multivector m defined by:

(*c*22) m: multi $(m, [[], 1, [2,3], [1,2,3]]);$

(*d*22) $[@(m\%_s, []), @(m\%_1, [1]), @(m\%_{2,3}, [2,3]), @(m\%_{1,2,3}, [1,2,3])]$

(*c*23) $u|m;$

(*d*23) $[@(m\%_1 u\%_1 g_{1,1}, []), @(u\%_1 m\%_s, [1]), @(u\%_2 m\%_s - m\%_{2,3} u\%_3 g_{3,3}, [2]),$
$@(u\%_3 m\%_s + u\%_2 g_{2,2} m\%_{2,3}, [3]), @(m\%_{1,2,3} u\%_3 g_{3,3} - m\%_1 u\%_2, [1,2]),$
$@(-m\%_1 u\%_3 - m\%_{1,2,3} u\%_2 g_{2,2}, [1,3]), @(u\%_1 g_{1,1} m\%_{1,2,3}, [2,3]),$
$@(u\%_1 m\%_{2,3}, [1,2,3])]$

(*c*24) $u.m;$

(*d*24) $[@(m\%_1 u\%_1 g_{1,1}, []), @(-m\%_{2,3} u\%_3 g_{3,3}, [2]), @(u\%_2 g_{2,2} m\%_{2,3}, [3]),$
$@(m\%_{1,2,3} u\%_3 g_{3,3}, [1,2]), @(-m\%_{1,2,3} u\%_2 g_{2,2}\ [1,3]), @(u\%_1 g_{1,1} m\%_{1,2,3}, [2,3])]$

(*c*25) $u \sim m;$

(*d*25) $[@(u\%_1 m\%_s, [1]), @(u\%_2 m\%_s, [2]), @(u\%_3 m\%_s, [3]), @(-m\%_1 u\%_2, [1,2]),$
$@(-m\%_1 u\%_3, [1,3]), @(u\%_1 m\%_{2,3}, [1,2,3])]$

(*c*26) $u \,\&\, m;$

(*d*26) $@(m\%_1 u\%_1 g_{1,1}, []).$

The examples given in (a) and (b) were chosen for their simplicity.

Of course the program performs the different products of more complicated multivectors.

6. The Derivative

The Geometric calculus allows derivation with respect to vectors, multivectors or scalars. Given a vector x:

(*c*17) $x;$

(*d*17) $[@(x\%_1, [1]), @(x\%_2, [2]), @(x\%_3, [3])].$

The procedure d constructs the derivative with respect to x:

(*c*24) $d[x];$

(*d*24) $$\left[@\left(\frac{d-}{dx\%_1}, [1]\right), @\left(\frac{d-}{dx\%_2}, [2]\right), @\left(\frac{d-}{dx\%_3}, [3]\right)\right],$$

$d[x]$ has the properties of a vector but can also act like a differential operator. If we consider only the vector properties of $d[x]$, all the operations described below can be used to construct various differential operators. For example: $d[x].x$ gives a scalar operator:

(*d*26) $$@\left(x\%_3 \frac{d-}{dx\%_3} + x\%_2 \frac{d-}{dx\%_2} + x\%_1 \frac{d-}{dx\%_1}, []\right),$$

$d[x].d[x]$ gives the operator:

$$\sum_{i=1}^{3} \left(\frac{\partial}{\partial x\%_i}\right)^2.$$

Now if we want $d[x]$ to act also like a differential operator on a given multivectorial quantity we must use doubled symbols for the products: $\|$, .., $\sim\sim$, &&. For example: $d[x] .. x$ gives 3 which is the divergence of x and $d[x] .. d[x]$ gives the usual Laplacian:

$$\sum_{i=1}^{3} \frac{\partial^2}{\partial x\%_i^2}.$$

A vector a will depend on the vector x if all its components depend on the components of x.

(*c*32) depend (a, x);

(*d*32) $[a\%(x\%_1, x\%_2, x\%_3)]$.

The divergence of a is computed by: $d[x] \,..\, a$ and the curl by: $d[x] \sim\sim a$.

The two operations are unified by the geometric product: $d[x] \,\|\, a$.

The last result is a multivector: $da = d..a + d \sim a$ (6.1) involving a scalar part $d.a$ and a bivector part $d \sim a$.

(*d*41)
$$\left[@\left(\frac{d}{dx\%_3}(a\%_3)+\frac{d}{dx\%_2}(a\%_2)+\frac{d}{dx\%_1}(a\%_1), []\right),\right.$$
$$@\left(\frac{d}{dx\%_1}(a\%_2)-\frac{d}{dx\%_2}(a\%_1), [1,2]\right),$$
$$@\left(\frac{d}{dx\%_1}(a\%_3)-\frac{d}{dx\%_3}(a\%_1), [1,3]\right),$$
$$\left.@\left(\frac{d}{dx\%_2}(a\%_3)-\frac{d}{dx\%_3}(a\%_2), [2,3]\right)\right].$$

The decomposition (6.1) may be recovered using the function GRADE and we can check that GRADE$(da, 0) \equiv d[x] \,..\, a$ and GRADE$(da, 2) \equiv d[x] \sim\sim a$.

The function d we used to construct the derivative with respect to a vector can also be used to construct a derivative with respect to a multivector. If follows the same principle. Given a multivector X:

(*c*66) x;

(*d*66) [@(x0, []), @(x1, [1]), @(x12, [1, 2]), @(x23, [2, 3]), @(x123, [1, 2, 3])].

$d[x]$ constructs a multivector which is also a differential operator (just like in the vector case):

(*d*64)
$$\left[@\left(\frac{d-}{dx0}, [1]\right), @\left(\frac{d-}{dx1}, [1]\right), @\left(-\frac{d-}{dx12}, [1,2]\right),\right.$$
$$\left.@\left(-\frac{d-}{dx23}, [2,3]\right), @\left(-\frac{d-}{dx123}, [1,2,3]\right)\right].$$

Now this operator may act on any multivector expression by means of the various products (using doubled symbols as in the vector case).

For example the expression $d[X] \,\|\, A$ computes the geometrical derivative of a multivector A with respect to the multivector X:

(*d*70)
$$\left[@\left(\frac{d}{dx1}(a\%_1), []\right), @\left(-\frac{d}{dx12}(a\%_2)+\frac{d}{dx0}(a\%_1), [1]\right),\right.$$
$$@\left(-\frac{d}{dx23}(a\%_3)+\frac{d}{dx0}(a\%_2)+\frac{d}{dx12}(a\%_1), [2]\right), @\left(\frac{d}{dx0}(a\%_3)+\frac{d}{dx23}(a\%_2), [3]\right),$$

$$@\left(-\frac{d}{dx123}(a\%_3)+\frac{d}{dx1}(a\%_2),[1,2]\right),@\left(\frac{d}{dx1}(a\%_3)+\frac{d}{dx123}(a\%_2),[1,3]\right),$$

$$@\left(-\frac{d}{dx123}(a\%_1),[2,3]\right),@\left(-\frac{d}{dx12}(a\%_3)-\frac{d}{dx23}(a\%_1),[1,2,3]\right)\Bigg].$$

The vector derivative is, of course, a particular case when the multivector X is homogeneous of grade 1.

Other useful operations such as reversion and dualization of multivectors (Hestenes & Sobczyk, 1984) have been also implemented.

6. Conclusions

In this paper only elementary operations on multivectors have been investigated. Despite its simplicity, our program is already a good tool to check and understand the basic rules of geometric calculus.

Using derivation with respect to a vector all the classical computations in vector analysis may be performed. Derivation of homogeneous multivectors with respect to a vector leads to a formalism directly related to the exterior calculus where exterior differentiation and coderivative of a p-form are defined.

Implementation of the derivative with respect to a multivector of a multivectorial expression opens the way to a generalization of the previous concepts.

References

Hestenes, D., Sobczyk, G. (1984). *Clifford Algebra to Geometric Calculus*. Dordrecht: Reidel.
Hestenes, D. (1966). *Space-Time Algebra*. New York: Gordon & Breach.
Hestenes, D. (1986). Curvature calculations with spacetime algebra. *Int. J. Theor. Phys.* **25**, 581–588.
Hsu, H. P. (1969). *Vector Analysis*. New York: Simon & Schuster.
MACSYMA (1985)—*Reference Manual*, Symbolics, Inc. Cambridge, MA: MIT.